BAS-VAS: A Novel Secure Protocol for Value Added Service Delivery to Mobile Devices

Neetesh Saxena *Senior Member, IEEE,* Mauro Conti *Senior Member, IEEE,*
Kim-Kwang Raymond Choo *Senior Member, IEEE,* Narendra S. Chaudhari *Senior Member, IEEE*

***Abstract*—Mobile operators offer a wide range of value-added services (*VAS*) to their subscribers (*i.e.*, mobile users), which in turn generates around 15% of the telecommunication industry revenue. However, simultaneous *VAS* requests from a large number of mobile devices to a single server or a cluster in an internet-of-things (*IoT*) environment could result in an inefficient system, if these requests are handled one at a time as the present traditional cellular network scenario is. This will not only slow down the server's efficiency but also adversely impacts the performance of the network. The current (insecure) practice of transmitting user identity in plaintext also results in traceability. In this paper, we introduce the first known protocol designed to efficiently handle multiple *VAS* requests at one time, as well as ensuring the secure delivery of the services to a large number of requesting mobile users. The proposed batch verification protocol (*BAS-VAS*) is capable of authenticating multiple simultaneous requests received by a large number of mobile users. We demonstrate that the protocol preserves user privacy over the network. The provider's servers ensure the privacy of the requested service's priority by performing sorting over encrypted integer data. The simulation results also demonstrate that the proposed protocol is lightweight and efficient in terms of communication and computation overheads, protocol execution time, and batch and re-batch verification delay. Specifically, we perform batch and re-batch verification (after detecting and removing malicious requests from the batch) for multiple requests in order to improve the overall efficiency of the system, as well as discussing time, space and cost complexity analysis, along with the security proof of our protocol using Proverif.**

***Index Terms*—Authentication, Batch Verification, Mobile User, Privacy Preservation, Value Added Service.**

I. Introduction

MOBILE computing and telecommunications industries are among the ones of the fastest growing industries worldwide. Mobile/cellular technologies, such as Long Term Evolution (*LTE*), are useful in environments where mobile users access Value-Added Services (*VAS*) through a Value-Added Service Provider (*VASP*). *VAS* are services offered by telecommunication operators to their subscribers for enabling direct, fast and easy access of information. As *VAS* are generally delivered over mobile devices, such services are also referred to as Mobile Value-Added Services (*M-VAS*). *M-VAS* are reportedly accessed by 75% of mobile subscribers at India in 2014 [1]. Popular *VAS* delivered via Short Message Service (*SMS*) include music and entertainment links and codes (*e.g.*, voting for reality shows and contests), booking of match and movie tickets, location based services, mobile advertising, *SMS* chatting and dating services, mobile banking through *SMS*, sports news and current affairs, weather reports, promote retail sales, festival related notifications, government reach to its citizen through *SMS*, and retrieving of game scores) [2]. For the foreseeable future and in the world of Internet-of-Things (*IoT*), *M-VAS* will have widespread applications. For example, *M-VAS* reportedly account for approximately 15% of telecommunication industry revenues [3]. The global *M-VAS* market is expected to increase to $655.07 billion by 2020 [4], and *M-VAS* market size is set to exceed *USD* 1,300 billion by 2024 [5]. However, balancing between security and efficiency in the service delivery to a large number of mobile devices will be challenging. According to Lerner *et al.* from Solon Telecoms, UK [6], among the *VAS* companies they have spoken with, most show strong continued organic growth in *SMS*-based revenues driven by growing penetration and ability to drive engagement. The cloud-based consumer *VAS* market revenue is estimated to be $48.4 billion in 2017 and is expected to reach $171.7 billion by 2023, growing at a *CAGR* of 23.5% during the forecast period 2017-2023. The *SMS* market revenue is estimated to reach $111.74 billion by 2023, growing at a *CAGR* of 21.5% during 2017-2023 [7].

Problem Statement: Operators of different cellular networks, such as Global System for Mobile Communication (*GSM*), Universal Mobile Telecommunications System (*UMTS*), and *LTE*, generally transmit data on the same channel as voice to optimize resource utilization, and non-voice data normally takes the form of *VAS* through *SMS* [8]. However, in Jan. 2014, the Guardian newspaper and Channel 4 News reported that the *NSA* collects (lawfully) and stores almost 200 million text messages per day across the globe. *NSA* programmes code named *"Dishfire"* and *"Prefer"* extracted location information, contacts and financial data from *SMS* messages, including automated texts, such as roaming charge alerts [9]. Vesselin mentioned on Security Tokens [10] that people are aware that *SMS*-based authentication is insecure because of security flaws like Singling System 7 (*SS7*).

N. Saxena is with the School of Computer Science and Informatics, Cardiff University, UK.
E-mail: nsaxena@ieee.org
M. Conti is with the Department of Mathematics, University of Padua, Italy.
E-mail: conti@math.unipd.it
K.-K. R. Choo is with the Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249 USA, also with the School of Information Technology and Mathematical Sciences, University of South Australia, Adelaide, SA 5001, Australia.
E-mail: raymond.choo@fulbrightmail.org
N. S. Chaudhari is with the Discipline of Computer Science and Engineering, Indian Institute of Technology, Indore, India.
E-mail: narendra@vnit.ac.in
Manuscript received June XX, 20XX; revised XXX XX, 20XX.

ITU-T Q.3615, which is a telecommunication standardization sector of International Telecommunication Union (*ITU*), provides the communication of location information between various Location-Based Services (*LBSs*) over *SMS* [11]. *ITU-T* X Series provides standard and guidelines for data networks, open system communications and security [12] (last updated on 4 June 2019). But at present, telecom operators do not implement any security services to *SMS* due to overhead and cost. More specifically, *ITU-T Rec. X.1146* identified the user identity authentication as a threat to *VAS* on single user account basis and does not discuss and support batch verification-based user authentication for *VAS*. According to a 2016 study [13], the usage of *SMS* for different services has many security flaws. The researchers have analyzed 400,000 text messages and found a significant portion of these messages was sent with confidential and private information (like credit card numbers, *CVV*, *PIN*, password reset options, etc.) that can be accessed over the network, as the traditional *SMS* does not contain any security, nor any authentication process for broadcast/multicast services. Also, in some countries, mobile operators may require a separate license for *M-VAS* [3]. Mobile operators such as Vodafone and Airtel provide *M-VAS* to their customers, based on user's information and usage preference [14], [15]. However, this may limit the design of suitable security protocols as well as affecting the performance of *M-VAS* delivery to a large number of mobile subscribers (in an *IoT* environment) and hence, user's quality of experience. For example, in the attempt to cater to wide ranging user information and usage preferences (*e.g.*, handling multiple *M-VAS* requests at any one time) could result in significant overheads, lengthy execution and verification times, and a bottleneck. The performance challenge is compounded when invalid authentication requests are transmitted to the *AS* due to half-open connections by flood-based Denial-of-Service (*DoS*) attacks and other cyber-attacks.

Saxena *et al.* proposed an "*EasySMS*" protocol for end-to-end secure transmission of *SMS* [16], but the protocol only deals with one *SMS* at a time. The system's performance degrades due to large overhead as there is an increase in the number of mobile users' requests for authentication. The *EasySMS* protocol generates 18, 36 (1 time 100% increase), 216 (11 times increase), and 2016 (111 times increase) bytes as the computation overhead when the number of mobile users' requests are 1, 10, 100, and 1,000, respectively. Similarly, this protocol produces 2152, 4456 (1.07 times increase), 27496 (11.77 times increase), and 257896 (118.84 times increase) bytes as the communication overhead for the same number of users' authentication requests. Clearly, single authentication-based *EasySMS* generates too much overhead and has performance issue with a large number of mobile users' authentication requests. Thus, in this paper, we focus on the computational effort required by the *VASP*, seeking to minimize wherever practical. In a real-world deployment, the communication bandwidth between a user and the *VASP* may be limited (*e.g.*, in rural areas with poor coverage). Therefore, we need to ensure the size of communicated messages to be as small as possible. In addition, services requested by a user should be un-linkable to each other, in order to ensure confidentiality of the user and *VAS*. The adversary may also target a specific user, based on the user's priorities for different services requested. For example, a user particularly in a sparsely populated area (*e.g.*, a rural town) could be identified or profiled due to the user's priorities for requested services. Therefore, it is important to hide the priority of the service requested by each user from the Authentication Server (*AS*), Service Providing Server (*SPS*) and adversary.

A. Existing Solutions

A number of solutions have been presented in the literature that are designed to provide *VAS* in different networks, such as Vehicular Ad-Hoc Networks (*VANETs*) [17], [18], [19], [20], public-private key based vehicular communication system [21], [22], privacy-preserving authentication and access control protocol in *VANET* [23], and social networks [24]. However, there is no known solution for batch-oriented *VAS* services to mobile users in a mobile/cellular network. In addition, while existing solutions generally are secure against Man-in-the-Middle (*MITM*) attacks, they do not provide integrity protection to the messages (with the exception in [20]). Zhang *et al.* [19] introduced a batch signature verification protocol *IBV*, based on identity-based cryptography, that can verify multiple received signatures at the same time. Zhang *et al.* [20] proposed a novel message authentication protocol named *RAISE* that adopts the k-anonymity property for preserving user privacy. They further proposed a supplementary protocol that can cooperatively work to probabilistically verify only a small percentage of these message signatures based on a device's computing capacity. Further, Huang *et al.* [18] introduced an anonymous batch authenticated and key agreement (*ABAKA*) protocol, based on Elliptic Curve Cryptography (*ECC*), to authenticate multiple requests sent from different vehicles and establish different session keys for different vehicles at the same time. Furthermore, Chim *et al.* [21] provided a software-based solution, *SPECS*, which uses bloom filter and trusted authority for key management and is based on pairing-based bilinear mapping. Horng *et al.* [22] found that *SPECS* is vulnerable to impersonation attack and hence, proposed an improved version named *b-SPECS+*.

We also remark that *IBV* in [19] does not deliver mutual authentication and only partially defeats impersonation attacks. While user privacy is ensured in *ABAKA* [18], *RAISE* [20] and *PAACP* [23], the *RAISE* protocol generates a large overhead and has a large storage requirement, as it maintains an *ID*-key table. The *PAACP* protocol is based on asymmetric key cryptosystem that also generates a large overhead. It provides access control to vehicles in order to communicate with road side equipment. Unlike *PAACP*, different *VAS* provided by many mobile operators do not impose any access restrictions to the users, as all *VAS* are publicly available to each mobile subscriber. Hence, in such a setting, an access control protocol is not required. Both *SPECS* [21] and *b-SPECS+* [22] protocols are vulnerable to replay attacks. There are a small number of wireless *SMS*-based protocols [25], [26], [27], but these protocols neither consider *VAS* nor handle multiple requests.

There are also a number of commercially available software such as *SMSzipper*, *TextSecure*, *moGile Secure SMS*, *CryptoSMS* that allow users to send secure *SMS*, but they are not suitable for *M-VAS*, as these software need to be regularly patched to avoid any disclosed vulnerabilities to be exploited. For example, *TextSecure* has already decided to remove the encryption function [28], and the subscribers will have to tap connection data for encrypted *SMS*. In fact, the developers want to get rid of the security feature due to the performance degradation and secret key distribution issue. Hence, we cannot completely build our trust on these apps, as some of their features or the applications itself can shut down at any time. Similarly, for example, *CryptoSMS* does not support all types of mobile devices [29]. Therefore, it is preferable to develop an authentication protocol (rather than a software) that would provide a secure communication interface between the service providers and the end users.

It is also important to follow the best practices on mobile device security. AlDairi *et al.* [30], for example, demonstrated several potential security and privacy attacks on mobile platform, such as eavesdropping, denial-of-service, man-in-the-middle, phishing, spoofing, data integrity, and identity theft. In order to defeat such attacks and achieve better awareness and security behavior understanding, Bitton *et al.* [31] introduced an expert-based procedure for deriving mobile security awareness models for different attack classes (each class is an aggregation of social engineering attacks that exploit a similar set of human vulnerabilities). Furthermore, Thompson *et al.* [32] evaluated data from 629 home computer and mobile device users to improve understanding of security behavior, and several factors particularly, the perceived vulnerability, self-efficacy, response cost, descriptive norm and psychological ownership were tested against both types of device users. Only perceived severity was only found to play a role in mobile device security behavior.

Generally, a single authentication (non-batch) refers to the authentication between a single mobile user and an authentication server at any one time. The server will not process any request sent by other mobile user until the first user's verification is successfully processed. On the other hand, in a batch authentication process, if malicious user requests are present, the process will not successfully complete. This will require the detection of invalid mobile user requests, remove them from the existing batch, and thereafter perform the batch authentication process again, such a process is also referred to as re-batch authentication. Batch authentication slightly differs from a group authentication. Generally, a group is (much) smaller than a batch, and a group is created for a specific purpose and a goal, in the sense that it has special properties, and group members work together towards the defined goal. On the other hand, a batch is simply an involvement and participation of several users belonging to one or several groups. There are no defined members for mobile users under a batch. A mobile user from any specific group can participate in a batch authentication at any time. In fact, a batch can contain several groups and their associated members. As an example, several mobile users with differing purposes (*e.g.*, dating-related services, e-ticketing services, and so on) can participate in this protocol (as long as the users interact with the same cellular service provider's authentication server) to secure their transmitted messages.

Though cellular-enabled data centers are the future, at present they have their own growing pains to overcome, and traditional *SMS*-based cellular systems are underway to implement this idea. As the prices of cellular data plans throughout the world continue to drop, the comparative cost of installing additional lines solely for the purpose of network fail over becomes more and more difficult to justify [33]. The *4G LTE* infrastructure already includes authenticated and encrypted internal signaling, integrity protection, enterprise end-point routers, devices with built-in firewall, and Virtual Private Network (*VPN*) tunneling with encryption for over the Internet connectivity. Data centers will be drawn to data plans featuring support for non-Internet-facing private *4G LTE* carrier networks as their traffic will be segregated from the Internet. In addition to the regular authentication of the mobile users, the network has to authenticate users asking for a specific *VAS*. For this second part, verification requires efficient handling of the requests of mobile users, as each user may opt for several *VAS* and can request many times in a day. Hence, authentication with a traditional process will be too much time consuming and the generated overhead will be way more than the authentication in a batch. Hence, we designed the *BAS-VAS* protocol that can manage the overall generated overhead and time efficiency of the verified requests.

To the best of our knowledge, there is no batch verification-based protocol (multiple user requests verification simultaneously) in the literature, which provides *M-VAS*, although there are many authentication protocols for *GSM* network [34], [35], [36], *UMTS* network [37], [38], [39], and *LTE* network [40], [41]. All these protocols are mainly used to achieve authentication, and do not execute authentication requests in a batch. Existing group authentication and key agreement protocols for *LTE* network [42], [43] are also not designed to provide *M-VAS* or based on *SMS*. Hence, such protocols are excluded in the performance evaluation. Recently in 2015, a solution was proposed to ensure mobile user privacy [44]. The solution considers that new International Mobile Subscriber Identity (*IMSI*) must always be chosen by the home network to prevent assigning a single *IMSI* to two different Universal Subscriber Identity Module (*USIM*) and suggests predefined multiple *IMSI* for each *USIM*. However, this solution requires a large storage, generates significant overhead for pseudo-identities, and consumes significant bandwidth to transmit various *IMSI* to each Mobile Subscriber (*MS*).

B. Our Contributions

In this paper (based on a preliminary idea published in [45]), we propose and present an efficient and secure batch oriented authentication and key agreement protocol, hereafter referred to as *BAS-VAS*, for providing *VAS* to mobile users. The protocol allows mobile users a fast and easy way of consuming *VAS*, as the protocol is based on symmetric key cryptography

and performs symmetric key encryption (with the exception of sorting encrypted messages), which is 1,000 times faster than asymmetric key encryption [46]. Moreover, the *AS* is capable of authenticating multiple requests at a time, which improves the overall efficiency of the system. The *BAS-VAS* enables efficient execution of *VAS* request with a lower verification delay. It also provides confidentiality and integrity to the messages, and non-linkability to previously received *VAS*. We regard the contributions of this work to be three-fold:

- Our proposed *BAS-VAS* provides mutual authentications between each *MS* and the *AS*. The *BAS-VAS* maintains confidentiality using Advance Encryption Standard with Counter mode (*AES-CTR*) and integrity using Message Authentication Code (*MAC*) function between both, *MS* and *AS*. These security features were not available in the earlier protocol [45].
- We propose and demonstrate the usefulness of keeping the original identity of each *MS* secret during its transmission over the network in mitigating *IMSI* capturing and impersonation attacks. In this approach, we do not need to generate multiple *IMSI* at the *AS* or send multiple *IMSI* to the *MS*, unlike in [44]. Moreover, unlike the protocol in [45], we use a standard *AES-CTR* to generate a Temporary Identity (*TID*) and an original *IMSI*.
- As the proposed protocol hides the priority of the service requested by each user from other users and servers, we present a protocol to perform sorting over the encrypted priorities of different services.

We then compare the performance of the proposed protocol with that of five protocols, namely: *ABAKA*, *SPECS*, *b-SPECS+*, *IBV* and *RAISE*. In a single/batch authentication, the *BAS-VAS* protocol achieves a reduction of 14.29% and 10% in transmission bandwidth from the device to the server, in comparison to both *ABAKA* and *RAISE*, respectively. However, transmission bandwidth of our protocol is only slightly larger than *IBV* [19], but *IBV* does not deliver mutual authentication and only partially defeats impersonation attack. Our protocol also reduces the communication bandwidth consumption from the server to the device by 60%, in comparison to *ABAKA* and *RAISE*, while *IBV* does not transmit data from the server to the device. In addition, our protocol uses less bandwidth than *SPECS* and *b-SPECS+* (35.23% and 69.42%, respectively).

The remainder of this paper is organized as follows: Section 2 presents the system and threat models, and the assumptions. Sections 3 and 4 present the proposed *BAS-VAS* protocol and its security analysis, respectively. The formal proof using Proverif is presented in Section 5. Sections 6 and 7 present the findings from the performance evaluation and simulation, respectively. Finally, Section 8 concludes this work.

II. Models and Assumptions

In this section, we present the system model (Section 2.1), threat model (Section 2.2), and system assumptions (Section 2.3).

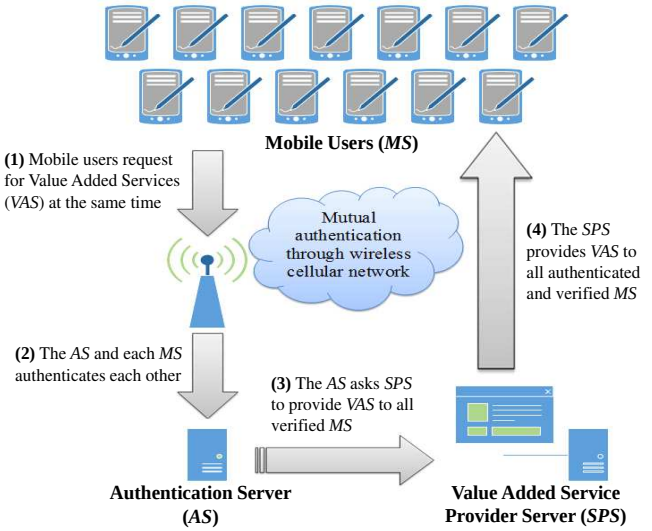


Fig. 1: Batch verification authentication requests for the *VAS*.

A. System Model

We consider a scenario for *VAS* where multiple *MS* send authentication requests to a *AS* simultaneously (or in fixed short time duration). The challenge is for the *AS* to verify and authenticate the maximum number of *MS* it is capable of handling efficiently – see Figure 1. Upon receiving the requests, *AS* will verify and send information of authenticated *MS* to the server responsible for the requested *VAS*. Thereafter, the *SPS* provides the service to all legitimate *MS*. These authentication requests may be single or multiple to a *AS*. However, in practice, it is unlikely to have only a single request at any one time. If the server is only capable of handling one request at a time, then a queue is required to manage all incoming requests. This will results in the additional task (*i.e.*, queue management, resulting in increased overheads, execution time, and costs of authentication). To scale well, the *AS* needs to be very efficient in handling a large number of requests sent in a burst (very short period of time).

One way to more efficiently handle multiple authentication requests simultaneously is to perform batch authentication for all incoming requests, and identify and remove invalid request (*e.g.*, generated by an adversary) prior to performing a re-batch authentication. While there are additional costs associated with the re-batch authentication, it can significantly reduce the overall authentication costs. We further extend our scenario in such a way where each *MS* can send its request with a priority ranging from 1 to 5 (1-least and 5-most important). This will allow the *SPS* to provide the required service according to the user's need (*e.g.*, user pays a premium to have the service delivered faster). The notations used in the remainder of this paper are presented in Table I.

B. Threat Model

In the *ideal model*, all mobile users compute the required functions in a probabilistic polynomial time, since the trusted *AS* is linked to all mobile users via a perfect private and

TABLE I: Notations

Symbol	Description	Size
IMSI	International mobile subscriber identity	128
TID	Temporary identity	128
G	Identity of service provider	128
SK	Shared secret key between *MS* and *AS*	128
DK	Delegation key generated from *SK*	128
MAC	Message authentication code	64
$T/T_i/T_1$	Timestamp	64
k	Random number	128
S	Token value generated by *MS*	128
Actcode	Activation code of *SK* key	64
$f_1()$	*HMACSHA256* is used to generate *DK*	–
$f_2()$	*AES-CTR* is used to generate *TID*	–
$f_3()$	*HMACSHA1* is used to generate *MAC*	–
$\oplus$	bitwise *XOR* operation	–

authenticated channel. All mobile users are also honest. In the *realistic model*, we consider a mix of mobile users, namely: honest majority, semi-honest majority and no-honest majority. The *honest majority* means that honest *MS* (legitimate to the network) and *AS* send correctly formed outputs to each other (*i.e.*, majority of the requests are legitimate), whereas for *semi-honest*, no more than half the *MS* send incorrectly formed outputs to the *AS* (*i.e.*, at least half of the requests are legitimate). In the *no-honest MS* (malicious *MS* to the network) scenario, more than half the *MS* send incorrectly formed outputs to the *AS* (*i.e.*, at most half of the requests are legitimate). Furthermore, malicious *MS* computes the required function in a probabilistic polynomial time with some arbitrary information. We do not consider these scenarios for the *AS*, as malicious *AS* does not have the correct keys in its database. Therefore, we consider only the trusted *AS* scenario, where the *AS* always sends correct outputs to all *MS*. All messages are sent from the *MS* to the *AS* (and vice versa) in time via an authenticated channel under the ideal model, while an adversary can choose to delay some, or all, messages under an unauthenticated channel in the realistic model. We assume that an authenticated channel provides end-to-end security to the transmitted message, and hence, there is no need for additional message encryption.

i) Adversary: Our threat model consists of a static (non-adaptive) and an adaptive adversary. In the static adversary threat model, the set of corrupted users are fixed. In the adaptive model, the adversary can corrupt any (number of) user(s) at will during run time. Furthermore, adversary can choose any input/output for the corrupted users. We also consider *passive and active adversaries* in the network.

ii) Security Attacks, and Integrity and Privacy Violations: The authentication protocol must provide *mutual authentication* [36], [39], where each *MS* must authenticate the *AS* to which it is requesting for a *VAS* and also the *AS* must verify the *MS*. Unidirectional authentication may lead to eavesdropping, redirection and impersonation attacks [39], [43]. Also, the half-open connection requests can be vulnerable to flood-based *DoS* attacks. In addition, the protocol needs to handle key generation, transmission and its usage. Specifically, the session key must not be sent over the network in plaintext. The original identity of each *MS* must also be protected during its transmission over the network. Such *privacy preservation* helps to prevent the system against *MITM* attacks [39], [44]. If *IMSI* is sent in cleartext, an adversary can target the system/user by tracing the user. Software such as *IMSI* catcher can be used to capture *IMSI* of a user over a weak or unencrypted network. Furthermore, *data confidentiality* and *message integrity* should be maintained in order to enhance the resilience of the system against common attacks [39]. Also, an adversary must not be able to link current session authentication information (*e.g.*, messages and keys) with previous sessions (*i.e.*, *non-linkability*), in addition to ensuring *forward/backward secrecy*.

C. System Assumptions

Traditional mobile system is generally implemented using symmetric key cryptosystem; thus, we choose a symmetric key cryptosystem with a lightweight protocol so that it is backward compatible (*i.e.*, can run on older mobile devices). However, we can add new features and services using any cryptographic primitives. We then use Paillier homomorphic encryption to implement sorting over encrypted integer data, which is not possible using symmetric homomorphic encryption in the considered scenario. Specifically, we consider the following system assumptions:

Assumption 1. each *MS* has a unique identity as *IMSI*[36].

Assumption 2. Authentication Center (*AuC*) is part of the *AS*; hence, *AS* $\subset$ Home Location Register (*HLR*). In *LTE*, the *HSS* (Home Subscriber Server) is the concatenation of the *HLR* and the *AuC*. The encryption at the Base Transceiver Station (*BTS*) works in the same way as in a traditional mobile network. The *AuC* sends a session key of a user to the *BTS* via secure channel [35], [37]. The *HLR* and *AuC* store information about a mobile subscriber. This information includes the *IMSI*, phone number, private key, and current location of the mobile user for packet and circuit switched operations.

Assumption 3. A secret key *SK* is stored in the *AuC*'s database as well as on the Subscriber Identity Module (*SIM*) of the *MS* at the time of manufacture, similar to a traditional mobile network [35], [37].

Assumption 4. The *AS* is a legitimate server, which does not disclose or send stored secret keys of one user to others without authorization, similar to a traditional mobile network [41].

III. Proposed Protocol: *BAS-VAS*

In this section, we present the proposed novel lightweight, efficient and secure batch-oriented protocol, *BAS-VAS*, for delivering *VAS* in a secure and timely manner (see Figure 3). *BAS-VAS* provides batch-oriented mutual authentication between the *AS* and all *MS*. The protocol also maintains message integrity between each *MS* and the *AS* using *MAC*,

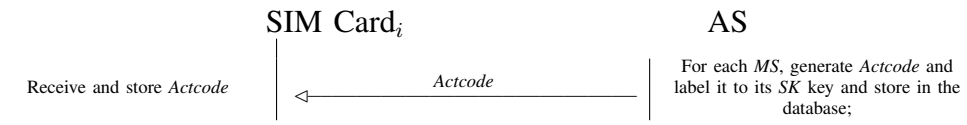


Fig. 2: User registration in the proposed protocol.

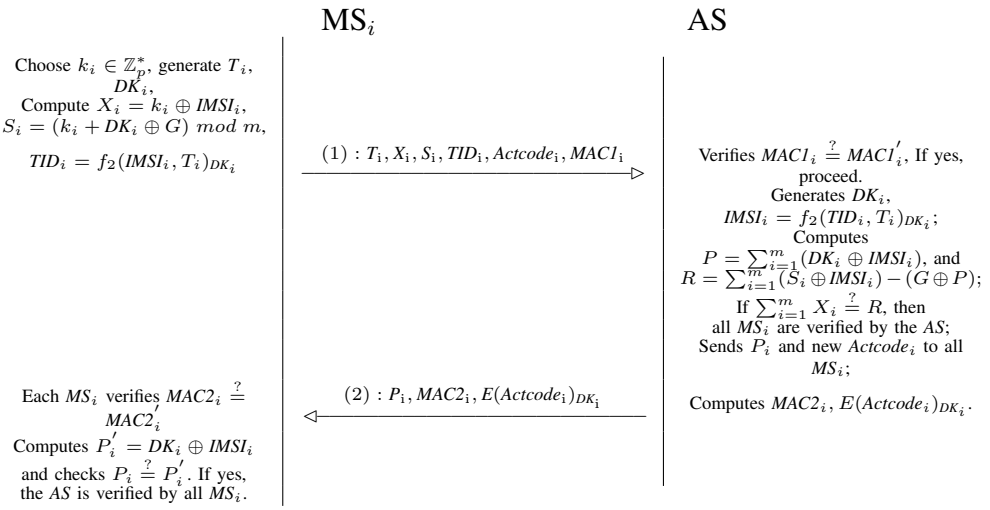


Fig. 3: Proposed *BAS-VAS* protocol.

which is an improvement of the protocol previously published in [45]. Let m be the total number of authentication requests generated by the mobile users MS_i (where i = 1, 2, 3, ..., m) and these requests are sent to the *AS* at the same time or in a burst. The value of m actually depends upon the capacity of the authentication server, which can then be publicly announced or privately communicated to each mobile subscriber as control information. The protocol comprises user registration, pseudo-identity creation, protocol initialization and protocol execution.

1. User Registration: Upon a user's request, the operator activates the *SIM* card by establishing a connection between the *SIM* card and the *AS*. The *AS* generates a random one-time code as *Actcode*, and stores *Actcode* in its database as a label to the secret key *SK*, prior to sending *Actcode* to the *SIM* card. The *SIM* card then receives and stores *Actcode* in the memory, as shown in Figure 2. When an *MS* requests for *VAS*, *Actcode* is sent from the *MS* to the *AS*. This allows the *AS* to retrieve the requesting user's *SK* key from the database without actually knowing the original user identity, *i.e.*, *IMSI*, rather just by receiving the *TID*, as shown in Figure 3, the *AS* side computations.

2. Pseudo-identity Creation: The generation of *TID* and retrieval of *IMSI* are not publicly available. We also introduce an encryption function to generate a temporary identity, where each MS_i generates a delegation key $DK_i = f_1(T_i)_{SK_i}$ derived from the shared secret key SK_i. Here, T_i is the current timestamp, and $f_1()$ a one-way hash-based *MAC* function (*e.g.*, *HMACSHA256* [47]). If the system is not synchronized or the use of timestamp is not practical, we can use a nonce (a random number) in place of timestamp value. Thereafter, each MS_i computes $TID_i = f_2(IMSI_i, T_i)_{DK_i}$ to avoid the need to transmit the original $IMSI_i$ over the network, as shown in Figure 3. This is designed to reduce the risk of *ID*-theft, eavesdropping and *MITM* attacks. We remark that the $f_2()$ function is just like any reversible symmetric encryption function, such as *AES* with counter mode (*AES-CTR*). The structure of this function may be known, but DK_i remains secret. This is another improvement over the protocol in [45], as in the latter, $f_2()$ was used as a non-standard function.

3. Protocol Initialization: Initially, each *MS* chooses a random number $k_i \in \mathbb{Z}_p^*$, generates the current timestamp T_i and a delegation key DK_i, where $\mathbb{Z}_p$ is a cyclic group of integers modulo p. In fact, this DK_i is generated at both, the MS_i and the *AS* using a shared secret key SK_i stored at the *AS* and on the *SIM* card at the time of manufacture. Thereafter, each MS_i computes $X_i = k_i \oplus IMSI_i$ and a token value $S_i = (k_i + DK_i \oplus G)\ mod\ m$, where $+$ is an addition and $\oplus$ is a bitwise-*XOR* operation. We refer to this token value a symmetric-signature. Here, each mobile user generates a valid symmetric-signature and fulfills the security properties with Assumption 4, namely: *authenticity* (signer signs associated message with its key), *unforgeability* (only signer can generate valid symmetric-signature for the associated message, while assuming *AS* a trusted entity), *non-reusability* (generated symmetric-signature cannot be used more than once), *non-repudiation* (signer cannot deny signing a previous message, *i.e.*, symmetric-signature with an assumption of the *AS* as a trusted entity), and *integrity* (ensures that contents have not been modified). In a symmetric key protocol, both parties (say, parties A and B) know the shared secret key. Hence, it would be challenging for a third-party receiving a message to determine whether the message was sent by party A or party B. Therefore, in our context, only the two parties are involved

and send messages to each other, since only both MS_i and *AS* know the corresponding secret key SK_i and the generated DK_i key.

When a message is sent from the MS_i to the *AS* with a token value (symmetric-signature) S_i, the *AS* knows that the used DK_i can only be generated by the corresponding MS_i (which can also be determined by verifying $IMSI_i$ from the received TID_i). Therefore, it is clear to the *AS* that the received message was actually sent by the corresponding MS_i. Similarly, this applies for the MS_i receiving a message from the *AS*. In addition, since the generated DK_i is not transmitted over the network, the adversary $\mathcal{A}$ is unable to compromise DK_i. One possible way is to provide *non-forgeability* and *non-repudiation* using asymmetric key cryptography, where each party sends an encrypted message signed using the sender's private key. On the other hand, the receiver verifies the digital signature using the sender's public key. Since *BAS-VAS* deals with only symmetric key cryptography, signatures generated by asymmetric keys are not performed. In the presented scenario, the adversary $\mathcal{A}$ cannot generate valid symmetric-signature, as it does not know the SK_i key and is unable to generate the DK_i key. Moreover, the MS_i and the *AS* are able to determine the sender of the message. Therefore, the process fulfills the required security properties.

4. Protocol Execution: **Step 1:** Each MS_i sends an authentication request, (T_i, X_i, S_i, TID_i) and $Actcode_i$, to the *AS* along with $MAC1_i = f_3(T_i, X_i, S_i, TID_i, Actcode_i)$, where $f_3()$ is *HMACSHA1* – a hash-based *MAC* function, and the key used is DK_i. On receiving the authentication requests, the *AS* first computes $MAC1_i^{'}$ and compares $MAC1_i \stackrel{?}{=} MAC1_i^{'}$. If the verification is successful, then the *AS* retrieves SK_i from the respective $Actcode_i$, and then computes DK_i and $IMSI_i = f_2(TID_i, T_i)_{DK_i}$. Next, the *AS* computes $P = \sum_{i=1}^{m}(DK_i \oplus IMSI_i)$ and $R = \sum_{i=1}^{m}(S_i \oplus IMSI_i) - (G \oplus P)$, where G is the identity of the service provider. If $\sum_{i=1}^{m}(X_i \stackrel{?}{=} R)$ at the *AS*, all MS_i are successfully verified by the *AS*. Otherwise, this implies one or more MS_i are malicious, which then requires a re-batch authentication.

Re-batch Authentication Process: In a re-batch authentication process, the *AS* first finds all invalid MS_i with the help of an algorithm in [45], [48]. Thereafter, the *AS* removes invalid MS_i from the batch and again computes $P = \sum_{i=1}^{m-t}(DK_i \oplus IMSI_i)$ and $R = \sum_{i=1}^{m-t}(S_i \oplus IMSI_i) - (G \oplus P)$, where t is the total number of invalid MS_i. The *AS* then compares $\sum_{i=1}^{m-t}(X_i \stackrel{?}{=} R)$. If it holds, all MS_i are authenticated by the *AS*. Otherwise, it repeats the re-batch authentication process.

A batch of authentication requests can be divided at most $\lceil log_2\ m \rceil$ times. At the end, this algorithm has a set of total number of invalid requests from MS_i and these invalid requests must be removed from the batch before a re-batch authentication takes place. Each invalid MS_i is placed in the black list of MS_i and can only be removed after a predefined time. During this period the request from particular MS_i is discarded.

Step 2: The *AS* sends all P_i to the respective MS_i along with $MAC2_i$ and $E(Actcode_i)_{DK_i}$, where $MAC2_i = f_3(P_i, E(Actcode_i)_{DK_i})$ and $E(Actcode_i)_{DK_i}$ is a randomly generated activation code encrypted by DK_i key. Each MS_i decrypts and uses $Actcode_i$ the next time it requests for *VAS*. On receiving the messages from the *AS*, all MS_i first compute $MAC2_i^{'}$ and compare $MAC2_i \stackrel{?}{=} MAC2_i^{'}$. If it holds, then all MS_i retrieve and store $Actcode_i$, compute $P_i^{'}$ and compare $P_i \stackrel{?}{=} P_i^{'}$, where $P_i^{'} = (DK_i \oplus IMSI_i)$. If the verification is successful, all MS_i verify the *AS*. Otherwise, the particular MS_i terminates the connection and initiates a fresh request to the *AS*.

Subsequent Authentication Request: Any subsequent request initiated by respective MS_i within the expiry time of DK_i is treated as a session request as shown in Figure 4 and is handled as follows:

Some of the computations (DK_i, X_i, S_i) are stored at the MS_i as well as at the *AS* until the expiry time of DK_i. Thus, if an MS_i requests for an authentication within the expiry time, then (DK_i, X_i, S_i) do not change at the MS_i. Instead, only a new TID_i is generated by the new T_i and $IMSI_i$. Each MS_i sends TID_i, T_i and $Actcode_i$ to the *AS*. Similarly, at the *AS*, only $IMSI_i$ is extracted from the received TID_i and T_i, after retrieving DK_i using $Actcode_i$. In such a scenario, DK_i remains the same within the session time. If the *AS* finds a valid MS_i session active, the *AS* sends $E(Actcode_i)_{DK_i}$ to the respective MS_i and asks the *SPS* to start delivering service to the corresponding MS_i.

Reliability of the protocol can be understood as follows [18], [45], [48]. In this protocol, if all MS_i are successfully verified then this protocol achieves its maximum reliability with respect to its performance. In such a case, this batch authentication protocol provides maximum successful authentications between the *AS* and MS_i at one time and generates minimum verification delay. Let N_{MS} be the maximum number of authentication requests generated at one point of time. Out of these requests, few can be invalid authentication requests, denoted as N_{IN}. Since, N_{MS} can be a very large number based on the type of value added service, the *AS* may not authenticate all the requests at one time due to its capacity. We assume

Algorithm 1 Invalid_req_algorithm (*AR*)

Input: The *AS* receives a batch (*AR*) of m authentication requests $\{R_1, R_2, R_3, \ldots, R_m\}$
Output: Returns the invalid request(s) otherwise return true

if (verify(*AR*)) **then** return True
else
  if (size(*AR*)==1) **then** return $IMSI_i \in AR$ as invalid request
  else
    set $AR_1 = \{R_1, R_2, R_3, \ldots, R_{\lceil m/2 \rceil}\}$
    set $AR_2 = \{R_{\lceil m/2 \rceil+1}, R_{\lceil m/2 \rceil+2}, R_{\lceil m/2 \rceil+3}, \ldots, R_m\}$
  Invalid_req_algorithm (AR_1)
  Invalid_req_algorithm (AR_2)

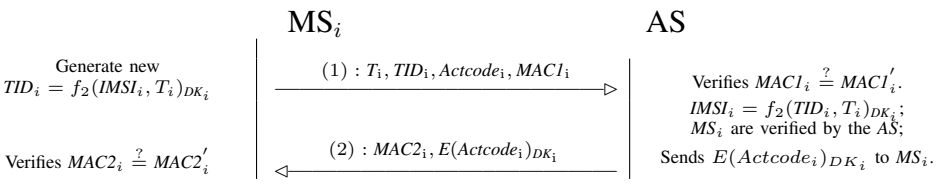


Fig. 4: Subsequent authentication request in *BAS-VAS* protocol.

that N_{AS} is the maximum capacity of the *AS* to authenticate the requests at one time. Let $Prob\{t\}$ be the probability that exactly t invalid authentication requests are sent to the *AS*. Then, the probability of the Hypergeometric distribution is as follows:

$$Prob\{t\} = \frac{\begin{pmatrix} N_{MS} - N_{IN} \\ N_{AS} - t \end{pmatrix}\begin{pmatrix} N_{IN} \\ t \end{pmatrix}}{\begin{pmatrix} N_{MS} \\ N_{AS} \end{pmatrix}}; \; t = 1, 2, \ldots, 10.$$

This indicates that (N_{AS} - t) valid requests are sent from (N_{MS} - N_{IN}). One or more invalid request(s) in a batch leads to batch verification failure and in such cases re-batch verification is required.

Impact of Mobility When a User Moves Out of Range of the AS: We also assume that various *AS* are installed and deployed at different geographic locations and are interconnected to each other with a pre-shared secret key between each pair of the *AS*. The *AS* where the user is registered is referred to as *Home-AS*, and all other *AS* deployed in roaming areas are *Visiting-AS*. When a mobile user requests for *VAS* in a roaming area, the corresponding *Visiting-AS* handles the request and sends the request message encrypted with a pre-shared secret key to the *Home-AS* of the user. The protocol execution takes place at the *Home-AS* and the result is securely provided to the *Visiting-AS*. Thereafter, the roaming *Visiting-AS* grants or revokes *VAS* to the user.

Distributing Different VAS to Respective SPS in Priority of User's Need: The *AS* authenticates all requests in a batch, irrespective of the type of service requested (ticket booking, news alerts, etc.). After successful authentication of the mobile users, the *AS* distributes the users' *VAS* requests and their encrypted priorities of the services to the respective *SPS*.

While this scenario is not a part of the authentication protocol, it provides a new feature of sorting encrypted integer priorities to ensure that the priority of the service requested by each user is not revealed to either *AS* or *SPS*. In such a case, we consider a public/secret keypair-based bilinear system for the *SPS*. Each MS_i encrypts a Priority Message ($E_{PK_{SPS}}\{PM_i\}$) using a public key of the *SPS* and sends the encrypted message to the *AS*. After authenticating all MS_i, the *AS* aggregates all requests grouped by different *VAS* and sends a list of TID_i and encrypted messages (discussed later) to the respective *SPS*, as shown in Figure 5. Upon receiving a list, each *SPS* performs sorting over the encrypted PM_i, arranges them in priority order as per users' need and notifies the *AS* that tells each user (by mapping TID_i with $IMSI_i$) about the waiting time of the requested service to be served. Subsequently, each *SPS*

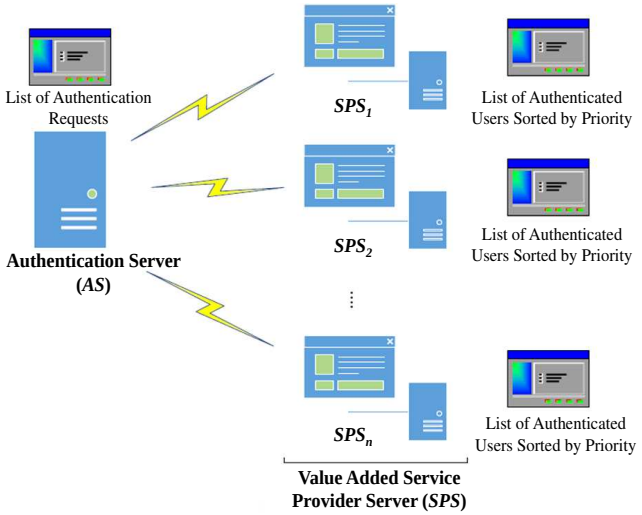


Fig. 5: An example scenario of *AS* sending a list of authenticated requests to each respective *SPS*.

provides its services to different users based on their order in the sorted list.

We use a Paillier homomorphic encryption scheme for this task, with an integer $N = p \times q$, where p and q are large primes [49]. Homomorphic encryption, denoted by [.], is used to encrypt the data represented by integers. A plaintext and a ciphertext are computed with $modulo\ N$ and $modulo\ N^2$. We use additive homomorphic encryption for two integers x and y, where $[x].[y] = [x+y]\ mod\ N^2$. The multiplicative inverse of $x\ modulo\ N^2$ is denoted by an integer $y = x^{-1}$, where $0 \leq y < N^2$ such that $xy = 1\ mod\ N^2$ and can also be used to negate an encrypted integer: $[x] \leftarrow [x]^{-1}\ mod\ N^2$.

We use a homomorphic encryption scheme [50] to perform additive operations in the following algorithms:

(a) *Key generation:* This algorithm generates the public keys and global parameters, given a security parameter. Let us consider two primes p and q, and $N = p \times q$. Choose a generator of the group $g \in \mathbb{Z}^*_{N^2}$. Let $\lambda(N) = lcm(p-1, q-1)$, where *lcm* represents least common multiple, and the public and secret keys of the receiver are *PK = (N, g)* and *SK =* $(\lambda(N))$, respectively.

(b) *Encryption:* The sender chooses message $M \in \mathbb{Z}_N$ and a random number $r \in \mathbb{Z}^*_{N^2}$. Thereafter, the sender computes the ciphertext using:

$$C = E(M) = g^M r^N\ mod\ N^2,$$

where r^N is used to generate different ciphertexts, even when the same message is encrypted twice.

(c) *Decryption:* In order to decrypt message C, the receiver computes:

$$M = D(C) = \frac{L(C^{\lambda(N)} \ mod \ N^2)}{L(g^{\lambda(N)} \ mod \ N^2)} \ mod \ N,$$

where the function L takes an input from a set $\{u < N^2 | u = 1 \ mod \ N\}$ and computes $L(u) = (u-1)/N$. In additive homomorphism, the sender computes two ciphertexts $C_1 = E(M_1)$ and $C_2 = E(M_2)$ from M_1, $M_2 \in \mathbb{Z}_N$, and the receiver decrypts $D\{(C_1.C_2 \ mod \ N^2) = E(M_1 + M_2) \ mod \ N\}$ that generates a sum of the plaintexts.

We now discuss the comparison and sorting of two encrypted *PM*, say [PM_1] and [PM_2]. Both *PM* have an upper bound d and the priority messages are PM_1/d and PM_2/d. The output is a bit, say t as $t = 1$ when $PM_1 \leq PM_2$ and zero otherwise. The relation with encrypted division as ($PM_1 \leq PM_2$) = $(d + PM_2 - PM_1)$/d.
If $PM_1 \leq PM_2$, then $x = d + PM_2 - PM_1 \geq d$ and $t = 1$; otherwise, $x < d$ and $t = 0$.

The steps of this process repeated for each pair of encrypted priority messages are as follows:

1) Upon receiving the encrypted priority messages from different MS_i, the *AS* chooses a random number r of $log_2 \ N - 1$ bits, encrypts r using *PK* key of the *SPS*, and computes $[x] \leftarrow [d + PM_2 - PM_1] = [d].[PM_2].[PM_1]^{-1}$ and $[z] \leftarrow [x + r] = [x].[r]$. Then, the *AS* sends $[r]$ and $[z]$ to the *SPS*.
2) The *SPS* decrypts $[r]$ and $[z]$ using its secret key (PR_{SPS}), and computes $r \ mod \ d$ and $z \ mod \ d$.
3) The *SPS* compares the inputs of PM_1 (*i.e.*, $r \ mod \ d$) and PM_2 (*i.e.*, $z \ mod \ d$) and observes the output. The output bit $t = 1$ for PM_1 iff $\{z \ mod \ d < r \ mod \ d\}$, and 0 otherwise.

We repeat the process for all unordered pairs of encrypted priority messages using Timsort [51]. This sorting algorithm takes advantages of partial orderings that exist in most real-world data. Baldimtsi *et al.* used Batcher sort, which sorts a set of n-elements using O($n \ (log \ n)^2$) data independent calls to a comparator function (*i.e.*, number of rounds is the same for a fixed n), where O($(log \ n)^2$) are consecutive levels and O(n) are pairs of elements compared and swapped at each level [52]. On the other hand, Timsort has O(n) best case complexity, and O($n \ log \ n$) average and worst case performance.

IV. Security Analysis

This section provides the security analysis of *BAS-VAS* based on various security aspects, such as prevention against security attacks, attempts to retrieve secret and delegation keys over the network, at the *AS* and at the *MS*, attempts to capture *IMSI* of the users, forward/backward secrecy, indistinguishability, and fairness and correctness of the protocol.

Property 1. *The proposed protocol defeats MITM, replay, redirection, flood-based DoS and impersonation attacks between the MS_i and the AS. Furthermore, the adversary $\mathcal{A}$ is not able to compromise the security of the message by delaying the message.*

BAS-VAS provides mutual authentication between the *AS* and the MS_i. Specifically, the *AS* authenticates the MS_i by checking $(\sum_{i=1}^{m} X_i) \stackrel{?}{=} R$, and each MS_i authenticates the *AS* by comparing $P_i \stackrel{?}{=} P_i^{'}$. Each MS_i receives $P_i = DK_i \oplus IMSI_i$ from the *AS* and computes its $P_i^{'} = DK_i \oplus IMSI_i$. If any of the MS_i or *AS* does not verify successfully, then the respective user terminates the connection. This process prevents our system against redirection and impersonation attacks, and also resolves flood-based *DoS* attack by performing re-batch verification. Furthermore, $f_2()$ is implemented as *AES-CTR* that hides the user's identity and encrypts the activation code each time the user requests for another *VAS*. Therefore, it helps to protect the system against *MITM* attacks, as $\mathcal{A}$ cannot capture *IMSI* using *IMSI* catcher. Furthermore, the timestamp (or nonce) values sent along with each message protect the system against replay attacks. Moreover, integrity protection of each transmitted message (message content and its threshold delivery in time, $T_{receive} \leq T_{generate} + T_{threshold}$) is maintained using *MAC* that prevents message tampering.

Property 2. *Adversary $\mathcal{A}$ is unable to extract SK_i and DK_i keys over the network, at the MS_i, and at the AS. $\mathcal{A}$ will not not be able to successfully retrieve the SK_i or DK_i key, even if it captures $Actcode_i$ of a mobile user sent over the network.*

A unique DK_i key is used within a session for each authentication between the *AS* and each MS_i. Each DK_i is generated from a SK_i key and is stored at the *AuC* and on the *SIM* card at the time of manufacture. Since a random $Actcode_i$ is sent over the network each time the MS_i requests for a *VAS*, the protocol is secure even if $\mathcal{A}$ is able to capture $Actcode_i$. Note that $\mathcal{A}$ cannot retrieve the SK_i and DK_i keys, as they are never sent over the network. Moreover, if $\mathcal{A}$ retrieves some $Actcode_i$, it cannot derive any relation among them, as these $Actcode_i$ are randomly generated. Moreover, each $Actcode_i$ is sent exactly once in plaintext over the network from the MS_i to the *AS*. The *AS* always generates a random $Actcode_i$ and sends to each MS_i as a ciphertext. Furthermore, if $\mathcal{A}$ modifies the encrypted $Actcode_i$ in message-2 from the *AS*, computed $MAC2_i^{'}$ will not match with the received $MAC2_i$ at the MS_i. Also, the message will not decrypt correctly using a modified or fabricated DK_i of the MS_i. Hence, the MS_i terminates the connection.

Property 3. *Adversary $\mathcal{A}$ cannot trace the original identity of the MS_i. In fact, $\mathcal{A}$ will also be unsuccessful in identifying the actual user, even if it captures the TID_i of a mobile user.*

Definition 1: (*Untraceability*): Our protocol satisfies untraceability, as $\mathcal{A}$ cannot distinguish whether two *TID* correspond to the same *MS* or two different *MS*.

$$Verify(publicChannel)[(IMSI_1, IMSI_2)|TID_i|MS|AS] \approx Verify(publicChannel)[IMSI_1|IMSI_2|TID_i|MS|AS].$$

Definition 2: (*IND-ANO: Indistinguishability under Anonymous Identity*): Our protocol is *IND-ANO*, as no adversary $\mathcal{A}$ at time t can distinguish between two chosen identities TID_1 and TID_2 with a negligible ϵ advantage.

$$Pr[\mathcal{A}(TID_1) = 1] - Pr[\mathcal{A}(TID_2) = 1] \leq \epsilon.$$

In our protocol, privacy of each MS_i is well protected. The TID_i is computed from original $IMSI_i$ as $TID_i = f_2(IMSI_i, T_i)_{DK_i}$. We implement $f_2()$ as *AES-CTR* with DK_i key, which is secure as no practical full attack has been found on *AES* at the time of this research. Furthermore, for all *VAS* requests including subsequent requests, a different TID_i is generated each time when a user connects to the *AS*. The *AS* flushes out TID_i from its memory, once a protocol run is completed. Hence, untraceability and identity anonymity are maintained, as $\mathcal{A}$ cannot trace TID_i to link with users and also $IMSI_i$ cannot be revealed to $\mathcal{A}$ and intermediate operators.

Property 4. *Adversary $\mathcal{A}$ cannot link current session information with the previous sessions. Moreover, our protocol maintains perfect forward/backward secrecy and chosen plaintext attack indistinguishability (IND-CPA) [53].*

For each fresh *VAS* request, the MS_i and the *AS* generate a fresh DK_i key with a unique timestamp, temporary identity of the user, $Actcode_i$ and k_i. Therefore, $\mathcal{A}$ cannot retrieve any information based on linkability among various *VAS* requests.

Definition 3: Our protocol maintains backward and forward secrecy, as no $\mathcal{A}$ could discover previously used session keys or generate future keys. $\mathcal{A}$ only wins if its output bit $b^{'}$ is equal to the random bit b in query and has a negligible advantage.

The SK_i and DK_i keys are never sent over the network. The DK_i is used to encrypt $IMSI_i$ and $Actcode_i$ using *AES-CTR*. Even compromising current DK_i will not allow $\mathcal{A}$ to generate past or future keys. Also, the past keys cannot be used for future sessions as both ends generate a new DK_i key.

Definition 4: (*IND-CPA: Indistinguishability under Chosen Plaintext Attack*): Our protocol is *IND-CPA* secure, as no adversary $\mathcal{A}$ in time t can distinguish between two chosen messages msg_1 and msg_2, and has no or negligible advantage.

$$\underset{DK_i \leftarrow SK_i}{Pr}[\mathcal{A}(msg_1) = 1] - \underset{DK_i \leftarrow SK_i}{Pr}[\mathcal{A}(msg_2) = 1] \leq \epsilon.$$

We assume that $\mathcal{A}$ has unlimited access to the encrypted data using a random oracle. In our protocol, the messages encrypted by the same key generate different ciphertext as at least one of the input parameters is always different. The MS_i generates TID_i as $f_2(IMSI_i, T_i)_{DK_i}$, where T_i changes each fresh message. Furthermore, the *AS* generates a fresh $Actcode_i$ for each new session and passes $Actcode_i$ to the respective MS_i as $E(Actcode_i)_{DK_i}$. Note, each fresh $Actcode_i$ is encrypted and sent over the network only once. We use *AES-CTR* as $f_2()$ that encrypts successive values of a counter with *AES*, and regurgitates concatenation of the encrypted blocks. *AES-CTR* stream never includes twice the same block and is *IND-CPA*.

Property 5. *The proposed protocol works well under both passive and active corruption attacks in the presence of static and adaptive adversaries $\mathcal{A}$. The protocol achieves fairness and guaranteed output delivery with "no MS_i (malicious or legitimate) having an advantage". The protocol maintains correctness under honest, semi-honest and no-honest majority scenarios.*

Passive corruption attack means that $\mathcal{A}$ obtains the complete information held by the corrupted MS_i; however, the MS_i still runs the protocol correctly. On the other hand, active corruption attack refers that $\mathcal{A}$ takes full control of corrupted MS_i. In both cases, our protocol works correctly, as it maintains *IND-CPA* indistinguishability as well as perfect forward/backward secrecy. Moreover, keys are never sent over the network and new delegation keys are generated for each session. Furthermore, both passive and active adversaries may be static (a set of corrupted MS_i is chosen before the protocol starts), or adaptive ($\mathcal{A}$ can choose any corrupted MS_i at any time during the run of the protocol). In any case, $\mathcal{A}$'s selection of corrupted MS_i does not affect our protocol.

A protocol is said to be fair if it ensures that no user can gain a significant advantage over other users, even if the protocol halts for any reason. Consider a scenario in which MS_1 and MS_2 communicate with the *AS* at the same time. If all the MS_1, MS_2 and *AS* are trusted, then the MS_1/MS_2 and the *AS* can learn each others information. However, the MS_1 cannot learn anything about MS_2's information and vice versa, as one user cannot obtain any other user's DK_i key. Also, the user cannot derive $IMSI_i$ and signature S_i of any other user, as DK_i is secret and k_i is randomly generated by the MS_i. Our protocol maintains *IND-CPA*; therefore, no MS_i has an advantage over others.

Our protocol works correctly under all three scenarios. We consider these scenarios at the MS_i only, whereas the *AS* is considered as a trusted server similar to the server in the traditional cellular network. This is because the *AS* keeps SK_i of all MS_i secret; therefore, it cannot be dishonest or semi-dishonest. The effectiveness of our protocol under all three scenarios can be observed using re-batch verification delay. Under these scenarios, our protocol maintains security properties, such as *IND-CPA*, forward/backward secrecy and fairness.

V. Performance Evaluation

This section evaluates the performance of the proposed protocol, in terms of transmission and computation overheads, execution time, and batch and re-batch verification times.

A. Communication Overhead

The communication overhead is defined as the total number of bits transmitted during the authentication over the network. Although in the literature we did not find any protocol that is directly related to our work (designed for batch verification and delivering of *VAS*), we compare the communication overhead of *BAS-VAS* with those of *ABAKA* [18], *SPECS* [21], *b-SPECS+* [22], *IBV* [19] and *RAISE* [20]. This is because these

protocols support mutual authentications while keeping the same flow of information. However, the verification delays differ between *VANET* protocols and mobile network protocols, such as *BAS-VAS*, because *VANET* networks have additional devices and road side equipment used to communicate within the network. We also discuss the transmission overhead during the single and batch authentication requests.

Single Authentication: In Table II, m = 1 in batch authentication denotes a single authentication process, and m is the maximum number of authentication requests generated by different mobile users at any one time. In *BAS-VAS*, the device is a mobile user and the server is an authentication server. It can be clearly observed that compared with *ABAKA* [18] and *RAISE* [19], *BAS-VAS* has a lower transmission overhead (*i.e.*, 72 and 32 bytes from the MS_i to the *AS* and the *AS* to the MS_i, respectively). However, this overhead is slightly larger, in comparison to *IBV* [19]. However, *IBV* does not transmit data from the server to the device. Moreover, *BAS-VAS* is more efficient than both *SPECS* [21] and *b-SPECS+* [22], as it generates only 114 bytes of communication overhead, whereas *SPECS* [21] and *b-SPECS+* [22] generate 176 and 288 bytes, respectively.

Batch Authentication: Multiple requests (m = 2, 3, ..., i) can be handled simultaneously by a batch verification process. We compare our protocol with other discussed protocols for a single (m = 1) as well as batch (m = 50, 100, 200, 500 and 1,000) authentications. In batch/single authentications from the device to the server, *BAS-VAS* achieves a 14.29% and 10% reduction in transmission bandwidth, in comparison to the *ABAKA* and *RAISE* protocols, respectively. Similarly, from the server to the device batch/single authentications, *BAS-VAS* reduces the bandwidth consumption by 60%, in comparison to the *ABAKA* and *RAISE* protocols, respectively. Moreover, *SPECS* and *b-SPECS+* use additional 35.23% and 60.42% of the bandwidth consumption as compared to *BAS-VAS*.

To show the practicability of the proposed protocol, we have compared the generated communication overhead of *BAS-VAS* with the *EasySMS* protocol [16] that supports only one-to-one (single) authentication. For m number of mobile users, the *EasySMS* generates 1896 + 256 $\times$ m, whereas the *BAS-VAS* creates 104 $\times$ m. When m = 1, 10, 100, and 1,000, the *BAS-VAS* outperforms *EasySMS* by 95.16% (2152-104/2152), 76.66% (4456-1040/4456), 62.17% (27496-10400/27496), and 59.67% (257896-104000/257896), respectively.

TABLE II: Communication Overhead in Batch Authentication: Comparative Summary

Protocol	Device to Server (bytes)	Intermediate Authority (bytes)	Server to Device (bytes)
ABAKA [18]	$84m$	–	$80m$
SPECS [21]	$48m$	$96m$	$32m$
b-SPECS+ [22]	$48m$	$176m$	$64m$
IBV [19]	$63m$	–	N/A
RAISE [20]	$80m$	–	$80m$
BAS-VAS	$72m$	–	$32m$

B. Computation Overhead

This section analyzes the computation overhead of *BAS-VAS* during a single as well as batch authentications.

Single MS_i Authentication with Active Session: The computed overheads at the MS_i and the *AS* are as follows:
At MS_i: $f_2(IMSI_i, T_i)_{DK_i}$, $f_1(T_i)_{SK_i}$, $k_i \oplus IMSI_i$, $Y = G \oplus DK_i$, $k_i + Y$, $DK_i \oplus IMSI_i$, $MAC1_i$, $MAC2_i^{'}$, $D(Actcode_i)_{DK_i}$.
At AS: $f_1(T_i)_{SK_i}$, $f_2(TID_i, T_i)_{DK_i}$, $DK_i \oplus IMSI_i$, $S_i^{'} = S_i \oplus IMSI_i$, $G^{'} = G \oplus P$, $S^{'} - G^{'}$, $MAC1_i^{'}$, $MAC2_i$, $E(Actcode_i)_{DK_i}$.

Batch MS_i Authentication with Active Session (i = 2, ..., m): The computed overheads at the MS_i and the *AS* are as follows:
At MS_i: $m[f_2(IMSI_i, T_i)_{DK_i}]$, $m[f_1(T_i)_{SK_i}]$, $m[k_i \oplus IMSI_i]$, $m[Y = G \oplus DK_i]$, $m[k_i + Y]$, $m[DK_i \oplus IMSI_i]$, $m[MAC1_i]$, $m[MAC2_i^{'}]$, $m[D(Actcode_i)_{DK_i}]$.
At AS: $m[f_1(T_i)_{SK_i}]$, $m[f_2(TID_i, T_i)_{DK_i}]$, $m[DK_i \oplus IMSI_i]$, $(m-1)[P = \sum_{i=1}^{m} P_i]$, $m[S_i^{'} = S_i \oplus IMSI_i]$, $G^{'} = G \oplus P$, $(m-1)[S^{'} = \sum_{i=1}^{m} S_i^{'}]$, $m[S^{'} - G^{'}]$, $(m-1)[\sum_{i=1}^{m} X_i]$, $m[MAC1_i^{'}]$, $m[MAC2_i]$, $m[E(Actcode_i)_{DK_i}]$.

The total computation overhead for a single authentication includes 4 *Enc/Dec* functions, 2 *key generation* functions, 6 *XOR*, 4 *MACs*, 1 *Addition* and 1 *Subtraction* operations. Similarly, the total computation overhead for a batch authentication includes $4m$ *Enc/Dec* functions, $2m$ *key generation* functions, ($5m$+1) *XOR*, $4m$ *MACs*, ($4m$-3) *Addition* and m *Subtraction* operations. One can observe that in the proposed protocol as well as the *EasySMS* protocol, the computation overhead is theoretically the same, *i.e.*, 18 bytes, when m (single authentication) = 1. The communication overhead generated by the proposed protocol is 104 bytes for single authentication, whereas the *EasySMS* protocol generates 2152 bytes. Clearly, the *BAS-VAS* protocol is more communication efficient than *EasySMS*.

VI. Simulation Findings

This section presents the simulation results of *BAS-VAS* in Java. We compute and evaluate the total execution time (Section VI.1), the batch verification time (Section VI.2) and the re-batch verification time (Section VI.3) of our protocol. We also present the time, space and cost complexity analysis of *BAS-VAS*.

A. Protocol Execution Time

This section describes the total execution time of the *BAS-VAS* protocol. We consider a client-server paradigm for the MS_i and the *AS*, the simulations are conducted on an Intel Core i3-2330M 2.20GHz machine with Windows7 and 256 MB RAM using *J2ME* with mobile emulator and *JDK1.7*. We simulate our protocol with 50 *MS* and a single *AS*. We

consider the average value of 30 iterations for each result. On average, the execution time to perform addition (T_{add}), *XOR* (T_{xor}) and subtraction (T_{sub}) are 0.000933 milliseconds (ms), 0.030322 ms and 0.000933 ms, respectively. On average, the server connection establishment time is 3383 ms, transmission time for message ($T_i, X_i, S_i, TID_i, MAC1_i, Actcode_i$) and message ($P_i, MAC2_i, E(Actcode_i)_{DK_i}$) are 8.35 ms and 9.49 ms, respectively. Table III and Table IV present the findings of computed parameters, where *Ext*, *MUP*, and *Enc* and *Dec* are execution time (ms), memory used by the program (bytes), and encryption and decryption operations (ms).

Furthermore, we implement $f_2()$ as *AES-CTR* with 128 bits DK_i key passing input as $IMSI_i/TID_i$ and T_i and receiving output of 128 bits [54]. We also implement *AES-CTR* for $Actcode_i$ encryption with input of 64 bits as $Actcode_i$ concatenated with 64 bits *FFFFFFFF*. Table III presents the findings obtained for $f_2()$ and *Actcode*. The encryption (generation of TID_i) took 13.6 ms and decryption (generation of $IMSI_i$) executed 4.2 ms for $f_2()$, while it took 12.8 ms and 4.1 ms, respectively, for the encryption and decryption of $Actcode_i$. The $f_1()$ and $f_3()$ are implemented as *HMACSHA256* and *HMACSHA1*, respectively. Table IV presents the findings, where the output of *HMACSHA1* and *HMACSHA256* are truncated to 64 and 128 bits as *MAC* and DK_i, respectively. The input to *HMACSHA1* and *HMACSHA256* are 512 bits.

The total execution time in a single authentication = server connection establishment time + transmission time for all messages + computation time at MS_i and *AS* = 4519.62 ms.

The total execution time in a batch authentication = 0.028456+4519.59$\times m$ ms.

The single authentication process takes 4.5 s, and takes 45 s, 451 s, and 4519 s, respectively, when 10, 100, and 1,000 mobile users are involved in the process. Overall, on average 4.5 s computation time will be required by each mobile user, which is reasonably good as the mobile phones can easily tolerate and perform this computation and handle the execution time. Hence, this overhead does not have any advert performance impact on real mobile phones and applications.

B. Verification Time

This section presents the verification time in the single and batch authentication requests. Here, verification time for an MS_i is the time between the sent message ($T_i, X_i, S_i, TID_i, MAC1_i, Actcode_i$) and received response message ($P_i, MAC2_i, E(Actcode_i)_{DK_i}$). The verification time computed by the *AS* is the time between the sent messages ($P_i, MAC2_i, E(Actcode_i)_{DK_i}$) and the protocol completion.

Verification time in a batch authentication:
For MS_i: 172$\times m$+13.6$\times m$+185$\times m$+185$\times m$+0.030322$\times$ (2m+1)+0.000933+0.000933$\times$(3m-3)+12.8$\times m$ = 0.028456+ 568.46$\times m$ ms.
For *AS*: 185$\times m$+0.030322$\times m$+4.1$\times m$ = 189.13$\times m$ ms.
Verification time in a single authentication:
Consider m = 1 in the above verification time.

TABLE III: Computations of $f_2()$ and *Actcode*

Function	ExT Enc (ms)	ExT Dec (ms)
$f_2() = AES\text{-}CTR$	13.6	4.2
$Actcode = AES\text{-}CTR$	12.8	4.1

TABLE IV: Computations of $f_3()$ and $f_1()$ functions

Function	ExT (ms)	MUP (bytes)
$f_3() = HMACSHA1$	172	1718448
$f_1() = HMACSHA256$	185	1718568

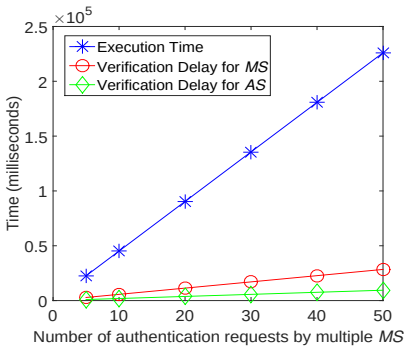
C. Re-batch Verification Time

If a batch authentication is not successful, then it is expected to execute a re-batch authentication without including the invalid MS_i. After detecting the invalid MS_i, we remove them from the batch and execute a re-batch authentication.

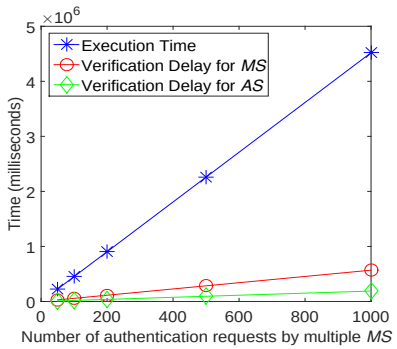
Total verification time in a re-batch authentication = 0.031255+ 0.002799$\times(m-t)$ ms = 0.000031+0.000002$\times(m-t)$ sec. This time is sufficiently small to be considered negligible.

Figure 6 depicts *BAS-VAS*'s execution time, verification time for the MS_i and verification time for the *AS*, when (a) the MS_i authentication requests m = 5, 10, 20, 30, 40, 50, and (b) m = 50, 100, 200, 500, 1,000. On average, for each *MS*, the protocol execution time, verification delay at the *MS* and verification delay at the *AS* are 4.5, 0.56 and 0.18 sec., respectively. Furthermore, Figure 7 represents a re-batch verification time, when m = 5, 10, 20, 30, 50; t = 1, 2, 4, 9, 25, and (b) m = 50, 100, 200, 500, 1,000; t = 1, 10, 20, 49, 99, 499. Note that the protocol execution time is the time in completing mutual authentication between all the MS_i and the *AS*, depending upon the number of authentication requests. For m = 50, the re-batch verification times are 0.17 ms and 0.10 ms, respectively, when t = 1 and t = 25 under honest majority and semi-honest majority scenarios. Similarly, for m = 1,000, the times are 2.8 ms and 1.4 ms, respectively, when t = 1 and t = 499 under honest majority and semi-honest majority. Under no-honest majority scenario, re-batch verification times are 0.079 and 0.033 ms, when m = 50 and t = 26 and 49, respectively. Similarly, for m = 1,000, the times are 1.029 and 0.033 ms, when t = 501 and 999, respectively.

In the scenario of an *IoT* environment where m = 100,000, the total execution time of all *MS* in a batch authentication is 451959000.02 ms. In other words, on average 4.519 s execution time will be required by each mobile user. Similarly, total verification times for *MS* and *AS* will be 0.568 s and 0.189 s (in total, 0.75 s), respectively, when m = 100,000. Total re-batch verification times when m = 100,000 will be 0.20 s (t =1), 0.199 s (t = 499), 0.199 s (t = 501), 0.198 s (t = 999), and 0.000033 s (t = 99999). Note that even m = 100,000, there is no observed adverse impact on the performance of the proposed protocol. In fact, the computed results remain consistent (changes were observed only in the decimal values).

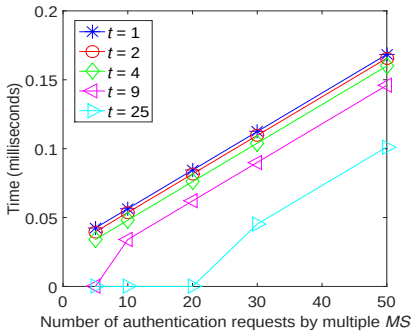


(a) m = 5, 10, 20, 30, 40, 50.

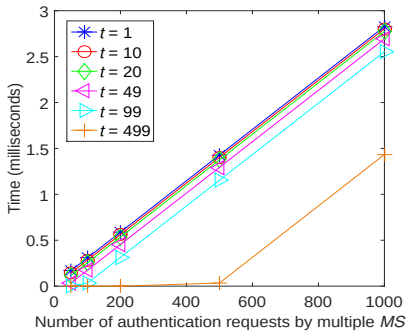


(b) m = 50, 100, 200, 500, 1,000.

Fig. 6: Execution time of *BAS-VAS* and verification delay for each *MS* and the *AS*, considering m authentication requests.



(a) m = 5, 10, 20, 30, 50; t = 1, 2, 4, 9, 25.



(b) m = 50, 100, 200, 500, 1,000; t = 1, 10, 20, 49, 99, 499.

Fig. 7: Re-batch verification time of *BAS-VAS*, considering t out of m authentication requests are malicious requests.

Thus, increasing the number of mobile users will not increase the protocol overhead and execution time. Hence, *BAS-VAS* supports scalability and efficiency, with adequate security.

D. Time, Space and Cost Complexity Analysis

In a single and batch authentication processes, we implement two functions $f_1()$ and $f_3()$ as *HMAC* functions, and *Actcode* and $f_2()$ as *AES-CTR*. The outputs of *HMACSHA1* and *HMACSHA256* are 160 bits and 256 bits, respectively. The DK_i key needs 128 bits from 160 bits and *MAC* requires 64 bits out of 256 bits. In total, 192 bits are stored. The time complexity for addition, subtraction and *XOR* operations are constant (*i.e.*, O(1)). For a single authentication (8 operations) and a batch authentication ($9\times m - 1$ operations), their cost are O(1). The block cipher algorithm (*e.g.*, *AES*) works with a fixed input size and has O(1) constant complexity. However, when the algorithm has a variable length input (say $|m|$), the time is O(m). For $f_2()$ and *Actcode* encryption in our protocol, the block size is fixed (128 bits) in *AES-CTR* (with random Initialization Vector (*IV*)). Therefore, the time complexity is independent of input and is constant O(1). Hence, in a single authentication (2 operations) and a batch authentication ($2\times m$ operations), the costs are O(1) for $f_2()$ and *Actcode* encryption/ decryption. The 128 bits of $IMSI_i$ and TID_i also need to be stored. Furthermore, storage for *HMACSHA1*, *HMACSHA256*, and *AES-CTR* at the MS_i and the *AS* are required. For a re-batch verification, O(1) is only the extra cost need to be paid (for $3m$-$3t$+2 operations). Therefore, *BAS-VAS* is an efficient, secure and cost effective protocol that requires less storage.

VII. Conclusion and Future Directions

The increasingly popularity of mobile devices and delivering *VAS* over mobile devices is a trend that is unlikely to go away any time soon. In this paper, we proposed a batch-oriented authentication and key agreement protocol that provides mutual authentication between each mobile user and the authentication server. The mutual authentication ensures the secure delivery of *VAS* to a legitimate requesting mobile user. Specifically, it efficiently verifies multiple requests sent by different *MS* at any one time while ensuring the original *IMSI* is kept private during the authentication as well as ensuring on-time delivery of *VAS*. Our protocol is also more efficient than the protocol in [44] in terms of preserving user privacy over the network. To the best of our knowledge, *BAS-VAS* is the first batch-oriented authentication protocol that provides *VAS* to mobile users. The protocol is designed on symmetric key cryptography, with the exception of our Paillier homomorphic encryption-based scheme. The latter scheme sorts encrypted integer data to maintain the privacy of the priorities of the requested services.

In both batch/single authentications from the device to the server, *BAS-VAS* achieves a significant reduction in transmission bandwidth by 14.29% and 10% respectively, in comparison to the *ABAKA* and *RAISE* protocols. From the server to the device in both batch/single authentications, the proposed protocol requires a lower (60%) communication bandwidth than the *ABAKA* and *RAISE* protocols. Moreover, our protocol is more efficient than *SPECS* and *b-SPECS+* by 35.23% and 60.42%, respectively. Findings from our Java simulations of the protocol indicated that the estimated re-batch verification time to be almost negligible, and in the worst case scenario, the re-batch verification time is 2.8 ms when one out of 1,000 requests is invalid (*i.e.*, 999 requests will need to be executed in a re-batch). The findings (*i.e.*, execution and verification times) also suggest the potential for our protocol to be deployed in a real-world mobile network.

Frequent mobility of users in their visiting networks is out of the scope of this work. However, in case (i) when the users belong to different geographical regions, their authentication will be achieved using the process discussed in the *EasySMS* protocol [16], and (ii) when the user moves to a visiting network, the authentication is completed by the process mentioned in *SAKA* [36], *ES-AKA* [39], and *IoT-enabled LTE AKA* [55], respectively, in *2G*, *3G*, and *4G* networks. The additional communication overhead in *BAS-VAS* will include routing transmitted information from the *Visiting-AS* to the *Home-AS*. The future direction of this work will include a lightweight implementation of a Homomorphic encryption and further reduction of overall computation and execution time.

APPENDIX A
FORMAL PROOF

This section presents the formal proof of the proposed protocol using Proverif, an automatic verification tool [56]. We perform five adversary queries: (i) Can an adversary successfully recover confidential information from the messages sent over the network?, (ii) Can an adversary successfully compute parameters generated by the *MS*?, (iii) Can an adversary successfully compute parameters generated by the *AS*?, (iv) Can an adversary successfully generate *DK* key of the *MS*?, and (v) Can an adversary successfully recover secret key of the *MS*? The following queries (under settings mentioned below) were made from an attacker point of view to verify whether the proposed protocol is secure:

```
(* The key table consists of pairs (ident , key) shared between
the MS and the HN. Table is not accessible by the attacker *)
table keys ( ident , key ) .
table keys2 ( ident , sessKey ) .
free s : bitstring [ private ] .

(* Secrecy Property *)
query attacker ( s ) .

(* The standard secrecy queries of ProVerif only *)
(* deal with the secrecy of private free names *)
(* DK is secret if and only if all DK are secret *)
free DK : sessKey [ private ] .
query attacker ( DK ) .
not attacker (new kims ) .

(* Authentication queries *)
event begAS( ident, sessKey ) .
event endAS( ident, sessKey ) .
event begMS( ident, sessKey ) .
event endMS( ident, sessKey ) .
query x1 : ident, x2 : sessKey ;
event (endAS( x1, x2 ) ) ==> event (begAS( x1, x2 ) ) .
query x1 : ident, x2 : sessKey ;
event (endMS( x1, x2 ) ) ==> event (begMS( x1, x2 ) ) .
event enableEnc .

(* When the attacker knows s, the event enableEnc has
been executed. *)
query attacker ( s ) ==> event ( enableEnc ) .
```

Run: The output from Proverif is as follows:

```
Neetesh@Neetesh-PC /proverif1.88
$ ./proverif examples/gsm/BAS-VAS.pv
Process: ( {1}!
(* Defining new values for MS. *)
{2}new imsims:ident;
{3}new kims:key;
{4}new tims:nonce;
{5}new actcodei:bitstring;
{6}new ki:bitstring;
{7}new G:bitstring;
{8}new m_36:bitstring;
{9}insert keys(imsims,kims);
{10}let DKms:sessKey = f1(tims,kims) in
{11}insert keys2(imsims,DKms);
(* Defining new functions with return
value for MS. *)
{12}let actcodei:bitstring = f5(rand) in
{13}let Xi:bitstring = f7(ki,imsims) in
{14}let Si:bitstring = f8(ki,DKms,G) in
{15}let Pi:bitstring = f9(DKms,imsims) in
{16}let tidms:ident = f2(imsims,tims,DKms)
in
{17}let mac1ims:mac = f3(tims,Xi,Si,tidms,
actcodei) in
{18}let mac2ims:mac = f4(Pi) in
{19}out(pubChannel,(MSG1,tims,Xi,Si,tidms,
mac1ims, actcodei));
{20}event begAS(imsims,DKms);
{21}in(pubChannel,(=MSG2,Pias:bitstring,
mac2ias:mac, sencrypt(actcodei,DKms)));
{22}if (mac2ims = mac2ias) then
{23}event endAS(imsims,DKms);
{24}in(pubChannel,(=CMC,enableEncms:bool));
{25}event endMS(imsims,DKms);
{26}in(pubChannel,(=MSG,msg:bitstring));
{27}out(pubChannel,sencrypt(msg,DKms));
{28}if (enableEncms = true) then
{29}let msgcontent:bitstring =
sdecrypt(msg,DKms) in 0 ) | (
(* Defining new values for AS. *)
{30}new kias:key;
{31}new DKas:sessKey;
{32}new ki2:bitstring;
{33}new Gas:bitstring;
{34}new mas:bitstring;
{35}in(pubChannel,(=MSG1,tims2:nonce,
Xi_38:bitstring,Si_39:bitstring,tidas:
ident,mac1ims2:mac,actcodei2: bitstring));
{36}let imsias:ident =
f2(tidas,tims2,DKas) in
{37}insert keys(imsias,kias);
{38}let DKas2:sessKey = f1(tims2,kias) in
{39}insert keys2(imsias,DKas2);
(* Defining new functions with return
value for AS. *)
{40}let actcodej2:bitstring = sdecrypt
(sencrypt(actcodei,DKms),DKms) in
{41}let Xias:bitstring = f7(ki2,imsias) in
{42}let Sias:bitstring = f8(ki2,DKas2,Gas)
in
```

```
{43}let Pias2:bitstring = f9(DKas2,imsias)
in
{44}let mac1ias:mac = f3(tims2,Xias,Sias,
tidas, actcodei2) in
{45}let mac2ias2:mac = f4(Pias2) in
{46}new msg_46:bitstring;
{47}if (mac1ims2 = mac1ias) then
{48}event endAS(imsias,DKas2);
{49}out(pubChannel,(MSG2,Pias2,mac2ias2));
{50}new enableEncas:bool;
{51}event begMS(imsias,DKas2);
{52}out(pubChannel,(CMC,enableEncas));
{53}out(pubChannel,sencrypt(msg_46,DKas2
));
{54}if (enableEncas = false) then
{55}event enableEnc;
{56}out(pubChannel,(MSG,s)) Else
{57}out(pubChannel,(MSG,sencrypt(s,
DKas2))))
```

```
Query attacker(s[]) ==> event(enableEnc)
Completing...ok, secrecy assumption
verified: fact unreachable attacker
(kims[!1 = v_945])
RESULT attacker(s[]) ==> event(enableEnc)
is true.
Query event(endMS(x1,x2)) ==>
event(begMS(x1,x2))
Completing...ok, secrecy assumption
verified: fact unreachable attacker
(kims[!1 = v_2079])
RESULT event(endMS(x1,x2)) ==>
event(begMS(x1,x2)) is true.
Query event(endAS(x1_2400,x2_2401)) ==>
event(begAS(x1_2400,x2_2401))
Completing...ok, secrecy assumption
verified: fact unreachable attacker
(kims[!1 = v_3256])
RESULT event(endAS(x1_2400,x2_2401)) ==>
event(begAS(x1_2400,x2_2401)) is true.
Query not attacker(DK[])
Completing...ok, secrecy assumption
verified: fact unreachable attacker
(kims[!1 = v_4331])
RESULT not attacker(DK[]) is true.
Query not attacker(s[])
Completing...ok, secrecy assumption
verified: fact unreachable attacker
(kims[!1 = v_5377])
RESULT not attacker(s[]) is true.
```

ACKNOWLEDGMENTS

Mauro Conti is supported by a Marie Curie Fellowship funded by the European Commission under the agreement PCIG11-GA-2012-321980. This work has been partially supported by the TENACE PRIN Project 20103P34XC funded by the Italian MIUR, and by the Project "Tackling Mobile Malware with Innovative Machine Learning Techniques".

REFERENCES

[1] The multi-billion dollar indian MVAS market opportunity, Beyond the Tip of the Iceberg (2014).
URL https://www.slideshare.net/wiprotechnologies/the-multibillion-dollar-indian-mvas-market-opportunity.

[2] 10 tips for the mobile value added services, Consult Strand.
URL http://www.strandreports.com/sw5420.asp.

[3] The department of telecommunications (DoT) India, Access Services.
URL http://www.dot.gov.in/access-services/list-access-service-licences-issued.

[4] Mobile value added services (MVAS) market worth $655.07 billion by 2020, markets and markets.
URL https://www.marketsandmarkets.com/PressReleases/mobile-value-added-service.asp

[5] MVAS market worth over $1,300 billion by 2024.
URL https://www.gminsights.com/pressrelease/mobile-value-added-services-mvas-market

[6] D. Lerner, A. Milovantsev, and M. Raghavan, emerging market value added services - a rare ray of light in the mobile industry (Dec. 2016).
URL www.solonstrategy.com/wp-content/uploads/2016/12/2016-12-06-Global-Trends-in-Mobile-VAS.pdf

[7] The mobile value-added services in cloud market revenue to reach $222.87 billion by 2023, Infoholic Research (Feb. 2018).
URL https://www.infoholicresearch.com/press-release/the-mobile-value-added-services-in-cloud-market-revenue-to-reach-222-87-billion-by-2023

[8] M. Ogul, S. Baktir, Practical attacks on mobile cellular networks and possible countermeasures, Future Internet 70 (2017) 474–489.

[9] US spy leaks: How intelligence is gathered, BBC (Jun. 2015).
URL www.bbc.co.uk/news/world-us-canada-24717495

[10] V. Bontchev, schneier on security (May 2019).
URL https://www.schneier.com/blog/archives/2019/05/on_security_tok.html

[11] ITU-T Q.3615, telecommunication standard sector of ITU, protocol for GeoSMS (Apr. 2015).
URL https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Q.3053-201703-I!!PDF-E&type=items

[12] ITU-T X Series, data networks, open system communications and security (last update on 4 June 2019).
URL https://www.itu.int/rec/T-REC-X/en

[13] B. Reaves, N. Scaife, D. Tian, L. Blue, P. Traynor, K. Butler, Sending out an SMS: Characterizing the security of the SMS ecosystem with public gateways, in: IEEE Symposium on Security and Privacy, 2016, pp. 339–356.

[14] A. Thakur, M. Kaur, H. Sharma, Subscriber behavior & mobile-value added services: a critical review, International Journal of Science Technology and Management 4 (2015) 125–132.

[15] Danish Khan, Bharti Airtel says content to play a larger role in its long-term strategy to drive data growth (2018).
URL https://telecom.economictimes.indiatimes.com/news/bharti-airtel-says-content-to-play-a-larger-role-in-its-long-term-strategy-to-drive-data/62804726

[16] N. Saxena, N. S. Chaudhari, EasySMS: A protocol for end-to-end secure transmission of SMS, IEEE TIFS 9 (7) (Jul. 2014) 125–132.

[17] X. Lin, X. Sun, P. H. Ho, X. Shen, GSIS: a secure and privacy preserving protocol for vehicular communications, IEEE TVT 56 (6) (2007) 3442–3456.

[18] J. L. Huang, L. Y. Yeh, H. Y. Chien, ABAKA: an anonymous batch authenticated and key agreement scheme for value-added services in vehicular ad hoc networks, IEEE TVT 60 (1) (2011) 248–262.

[19] C. Zhang, R. Lu, X. Lin, P. H. Ho, X. Shen, An efficient identity based batch verification scheme for vehicular sensor networks, in: INFOCOM, IEEE, 2008, pp. 816–824.

[20] C. Zhang, X. Lin, R. Lu, P. H. Ho, X. Shen, An efficient message authentication scheme for vehicular communications, IEEE TVT 57 (6) (2008) 3357–3368.

[21] T. W. Chim, S. M. Yiu, L. C. K. Hui, V. O. K. Li, SPECS: secure and privacy enhancing communications schemes for VANETs, Ad Hoc Networks 9 (2011) 189–203.

[22] C. Zhang, X. Lin, R. Lu, P. H. Ho, X. Shen, batch verification for secure pseudonymous authentication in VANET, IEEE Transactions on Information Forensics and Security 8 (11) (2013) 1860–1875.

[23] L. Y. Yeh, Y. C. Chen, J. L. Huang, PAACP: A portable privacy-preserving authentication and access control protocol in vehicular ad hoc networks, Computer Communications 34 (2011) 447–456.

[24] L. Y. Yeh, Y. L. Huang, A. D. Joseph, S. W. Shieh, W. J. Tsaur, A batch-authenticated and key agreement framework for P2P-based online social networks, IEEE TVT 61 (4) (2012) 1907–1924.

[25] J. L. Lo, J. Bishop, J. H. Eloff, SMSSec: an end-to-end protocol for secure SMS, Computers & Security 27 (5) (2008) 154–167.

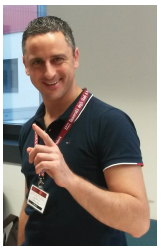
[26] H. Rongyu, Z. Guolei, C. Chaowen, X. Hui, Q. Xi, Q. Zheng, A PK-SIM card based end-to-end security framework for SMS, Computer Standards and Interfaces 31 (4) (2009) 629–641.

[27] C. C. Yang, Y. L. Tang, R. C. Wang, A secure and efficient authentication protocol for anonymous channel in wireless communications, Applied Mathematics & Comput. 169 (2) (2005) 1431–1439.

[28] F. A. Scherschel, Krypto-Messenger: "TextSecure" ohne verschlsselte SMS, "Sicher" jetzt sicherer (Jul. 2014).
URL https://www.heise.de/security/meldung/Krypto-

Messenger-TextSecure-ohne-verschluesselte-SMS-Sicher-jetzt-sicherer-2269353.html
[29] CryptoSMS device list, CryptoSMS.org (2010). URL http://cryptosms.org/devices.html
[30] A. AlDairi, L. Tawalbeh, Cyber security attacks on smart cities and associated mobile technologies, Procedia Computer Science 109C (2017) 1086–1091.
[31] R. Bitton, A. Finkelshtein, L. Sidi, R. Puzis, L. Rokach, A. Shabtai, Taxonomy of mobile users' security awareness, Computers & security 73 (2018) 266–293.
[32] N. Thompson, T. J. McGill, X. Wang, Security begins at home: determinants of home computer and mobile device security behavior, Computers & security 70 (2017) 376–391.
[33] T. Merenda, cellular in the enterprise data center (Jul. 2014). URL https://www.networkcomputing.com/wireless-infrastructure/cellular-enterprise-data-center
[34] C. C. Lo, Y. J. Chen, Secure communication mechanisms for GSM networks, IEEE Transactions on Consumer Electronics 45 (4) (1999) 1074–1080.
[35] C. C. Chang, J. S. Lee, Efficient authentication protocols of GSM, Computer Comm. 28 (8) (2008) 921–928.
[36] N. Saxena, N. S. Chaudhari, SAKA: a secure authentication and key agreement protocol for GSM networks, CSI Transactions on ICT 1 (4) (2013) 331–341.
[37] M. Zhang, Y. Fang, Security analysis and enhancements of 3GPP authentication and key agreement protocol, IEEE Transactions on Wireless Communication 4 (2) (2005) 734–742.
[38] Y. L. Huang, C. Y. Shen, S. W. Shieh, S-AKA: a provable and secure authentication key agreement protocol for UMTS networks, IEEE TVT 60 (9) (2011) 4509–4519.
[39] N. Saxena, J. Thomas, N. S. Chaudhari, ES-AKA: an efficient and secure authentication and key agreement protocol for UMTS networks, Wireless Personal Communications 84 (3) (2015) 1981–2012.
[40] F. Hadiji, F. Zarai, A. Kamoun, Authentication protocol in fourth generation wireless networks, in: WOCN, IEEE, 2009, pp. 36–39.
[41] G. M. Koien, Mutual entity authentication for LTE, in: IWCMC, 2011, pp. 689–694.
[42] Y. W. Chen, T. Wang, K. H. Chi, C. C. Tseng, Group-based authentication and key agreement, Wireless Personal Communications 62 (4) (2012) 965–979.
[43] C. Lai, H. Li, R. Lu, X. Shen, SE-AKA: a secure and efficient group authentication and key agreement protocol for LTE networks, Computer Networks 57 (17) (2013) 3492–3510.
[44] M. S. A. Khan, C. J. Mitchell, Improving air interface user privacy in mobile telephony, in: International Conference on Research in Security Standardisation, 2015, pp. 165–184.
[45] N. Saxena, N. S. Chaudhari, VAS-AKA: an efficient batch verification protocol for value added services, in: SMC, IEEE, 2013, pp. 1560–1565.
[46] C. C. Yang, Y. L. Tang, H. W. Yang, A secure and efficient authentication protocol for anonymous channel in wireless communications, Applied Mathematics & Comput. 169 (2) (2005) 1431–1439.
[47] HMACSHA256 (2013). URL https://msdn.microsoft.com/en-us/library/system.security.cryptography.hmacsha256(v=vs.110).aspx.
[48] N. Saxena, H. Shen, N. Komninos, K.-K. R. Choo, N. S. Chaudhari, BVPSMS: a batch verification protocol for end-to-end secure SMS for mobile users, IEEE TDSC,doi:10.1109/TDSC.2018.2799223.
[49] T. Veugen, Encrypted integer division and secure comparison, Int. J. Applied Cryptography 3 (2) (2014) 166–180.
[50] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: EUROCRYPT, 1999, pp. 223–238.
[51] Timsort, Java code. URL http://cr.openjdk.java.net/ martin/webrevs/openjdk7/timsort/raw_files/new/src/share/classes/java/util/TimSort.java.
[52] F. Baldimtsi1, O. Ohrimenko, Sorting and searching behind the curtain, in: Financial Cryptography and Data Security, Vol. 8975, 2015, pp. 127–146.
[53] Security guidelines for cryptographic algorithms in the W3C web cryptography API. URL http://www.w3.org/2012/webcrypto/draft-irtf-cfrg-webcrypto-algorithms-00.
[54] N. Saxena, N. S. Chaudhari, SecureSMS: a secure SMS protocol for VAS and other applications, JSS 90 (2014) 138–150.
[55] N. Saxena, S. Grijalva, N. S. Chaudhari, Authentication protocol for an IoT-enabled LTE network, ACM TOIT 16 (4) (Dec. 2016) 1–20.
[56] ProVerif: Cryptographic protocol verifier. URL http://prosecco.gforge.inria.fr/personal/bblanche/proverif

Neetesh Saxena is an Assistant Professor in Cyber Security at Cardiff University, UK. Before joining to CU, he was an Assistant Professor (Lecturer) at Bournemouth University, UK. Prior to this, he was a Postdoctoral Researcher at the Georgia Institute of Technology, USA and the State University of New York Korea, and a Visiting Scholar at the Stony Brook University, USA. He earned his Ph.D. in Computer Science & Engineering from Indian Institute of Technology (IIT), Indore, India. He has published several papers in international peer-reviewed journals and conferences. He was a DAAD and TCS Research Fellow and currently is a senior member of the IEEE and an ACM member.



Mauro Conti is a Full Professor at the University of Padua, Italy. He obtained his Ph.D. from Sapienza University of Rome, Italy, in 2009. After his Ph.D., he was a Post-Doc Researcher at Vrije Universiteit Amsterdam, The Netherlands. In 2011 he joined as Assistant Professor the University of Padua, where he became Associate Professor in 2015. He has been Visiting Researcher at GMU, UCLA, UCI, TU Darmstadt, UF, and FIU. He is a recipient of the Marie Curie Fellowship as well as the DAAD fellowship. He published more than 180 papers in top international peer-reviewed journals and conference. He is Associate Editor for several journals, including IEEE COMST and TIFS. He was Program Chair for TRUST'15, ICISS'16, WiSec'17, and General Chair for SecureComm'12 and SACMAT'13. He is a Senior Member of the IEEE.

Kim-Kwang Raymond Choo is the holder of the cloud technology endowed professorship at University of Texas at San Antonio, an associate professor at University of South Australia, and a guest professor at China University of Geosciences. He was named one of 10 Emerging Leaders in the Innovation category of The Weekend Australian Magazine/Microsoft's Next 100 series in 2009, and is the recipient of various awards including ESORICS 2015 Best Paper Award, Winning Team of Germany's University of Erlangen-Nuremberg Digital Forensics Research Challenge 2015, 2014 Highly Commended Award by Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and 2008 British Computer Society's Wilkes Award. He is a Fellow of the Australian Computer Society.

Narendra S. Chaudhari has completed his undergraduate, graduate, and doctoral studies at IIT, Mumbai, India, in 1981, 1983, and 1988 respectively. He has shouldered many senior level administrative positions, a few notable assignments include: Dean - Faculty of Engineering Sciences, Devi Ahilya University, Indore, Coordinator - International Exchange Program, NTU Singapore, and Dean - R&D, IIT Indore. He has more than 300 publications in top quality international conferences and journals. Currently, he is vice chancellor of the Uttarakhand Technical University, India and Professor IIT Indore, India. He is a Fellow and recipient of Eminent Engineer award (Computer Engineering) of IE-India, as well as Fellow of IETE-India, and Senior Member of CSI and IEEE.